Mathematical Biosciences and Engineering

Mathematical Biosciences and Engineering, 16(4): 2063–2085.
DOI: 10.3934/mbe.2019101
Received: 11 December 2018
Accepted: 19 February 2019
Published: 11 March 2019

http://www.aimspress.com/journal/MBE

Research article

An integrated approach for remanufacturing job shop scheduling with routing alternatives

Lingling Li[1], Congbo Li[2,*], Li Li[1], Ying Tang[3] and Qingshan Yang[2]

[1] College of Engineering and Technology, Southwest University, Chongqing, 400715, China
[2] State Key Laboratory of Mechanical Transmission, Chongqing University, Chongqing 400044, China
[3] Department of Electrical and Computer Engineering, Rowan University, Glassboro, NJ 08028, USA

* **Correspondence:** Email: congboli@cqu.edu.cn.

Abstract: Remanufacturing is a practice of growing importance due to increasing environmental awareness and regulations. However, the stochastic natures inherent in the remanufacturing processes complicate its scheduling. This paper undertakes the challenge and presents a remanufacturing job shop scheduling approach by integrating alternative routing assignment and machine resource dispatching. A colored timed Petri net is introduced to model the dynamics of remanufacturing process, such as various process routings, uncertain operation times for cores, and machine resource conflicts. With the color attributes in Petri nets, two types of decision points, recovery routing selection and resource dispatching, are introduced and linked with places in CTPN model. With time attributes in Petri nets, the temporal aspect of recovery operations for cores as well as the evolution dynamics in cores' operational stages is mathematically analyzed. A hybrid meta-heuristic algorithm embedded scheduling strategy over CTPN is proposed to search for the optimal recovery routings for worn cores and their recovery operation sequences on workstations, in minimizing the total production cost. The approach is demonstrated through the remanufacturing of used machine tool and its effectiveness is compared against another two cases: baseline case with fixed recovery process routings and case 2 using standard SA/MST.

Keywords: remanufacturing; job shop; scheduling; colored timed petri nets; simulated annealing

1. Introduction

Remanufacturing, as an industrial process of restoring discarded products/components back to their useful lives, is of growing importance due to the emerging pressure of legislation and increasing awareness of environmental conservation. As an ultimate form of recycling, remanufacturing maintains much of the value added from original material and reprocessing conservation, leading to lower production costs and improved profits [1]. A recent wall street journal revealed that large scale remanufacturing in the United States employs more than 500,000 people and contributes to approximately $100 billion of goods sold each year [2]. More successful industry applications can be seen in automobile remanufacturing, aerospace remanufacturing and electronic remanufacturing [3–5].

Compared to manufacturing, remanufacturing is more complicated in the way that (1) the supply of returned products is unpredictable in timing and quantities; (2) the quality and composition of returned products varies; and (3) the process routings are not necessarily fixed but rather adapt to the actual conditions of products/components [6]. Such characteristics have led many new methodologies to deal with various operation management issues in remanufacturing as summarized in the survey literature [7]. One of the challenges, which is the focus of this paper, is remanufacturing job shop scheduling. With the high level of uncertainty and resource limitation, it becomes extremely difficult to carry out various remanufacturing tasks for different products and still achieve maximal system performance.

Guide and his colleagues are the first group of researchers that looked into the remanufacturing scheduling. Through simulation, they examined the performance of several static dispatching rules in a repair shop. They assumed the system has stochastic arrivals, exponentially distributed process times and probabilistic process routings [8]. Extended from this work, an analytical model was further developed to explore the impact of those rules on system performance in terms of weighted average sojourn time [9]. While the pursued simulation and model analysis provided insights to practicing managers in a repair shop, much of work is needed for complex job shop settings where various products/parts with uncertain remanufacturing routings compete for constrained resources.

A second line of work has focused on economic lot scheduling with random returns. For instance, Tang and Teunter considered a hybrid system where (re)manufacturing operations of multiple products are performed on the same production line [10]. The work formulated the problem as a mixed integer linear program to determine the optimal cycle time and production starting times for (re)manufacturing. The research was further extended to relax the constraint of common cycle time and a single (re)manufacturing lot per item per cycle [11]. By dividing the returns into different quality grades, Sun et al. investigated a lot scheduling problem where the remanufacturing rate increases as the quality grade increases and holding costs for serviceable products are higher than returns [12]. The objective is to minimize the average total cost by optimizing the acquisition lot size and scheduling the remanufacturing sequence. However, the deterministic nature of the assumptions (i.e., constant demand and returns, and constant remanufacturing rate) made in those works failed to address the uncertainty inherent in remanufacturing. Moreover, the authors considered remanufacturing as a single-stage process. Given that refurbishing used products up to like-new condition tends to involve a sequence of recovery operations, such single-stage scheduling methods present practical limitation.

A third line of researches have tackled the multi-stage and multi-product remanufacturing scheduling problem. For instance, Kim et al. considered a remanufacturing system with a single

disassembly workstation, parallel flow-shop-type reprocessing lines, and a single reassembly workstation [13]. A hybrid scheduling heuristic is proposed to determine the sequences of products to be disassembled, to be reprocessed at each reprocessing lines and to be reassembled. The authors, however, treated the problem deterministic and static. Luh et al. presented another work that considered a similar remanufacturing system but tried to address the underlying uncertainty [14]. The main novelty of the research is that it considered stochastic asset arrival time and part repair time to model the dynamics of rotable inventory for optimization. Nevertheless, the authors presumed that each asset goes through a fixed series of overhaul operations. Giglio et al. studied a production system which produces multi-class single-level products through both manufacturing of raw materials and remanufacturing of return products [15]. A mixed-integer programming formulation is proposed to obtain the optimal lot size and job shop schedules with minimum total cost. This work presented the same limitation that the recovery process routings of returns are deterministic. By considering the alternatives of remanufacturing process routes for worn cores, Zhang et al. proposed a simulation based genetic algorithm approach for remanufacturing process planning and scheduling, where the processing time of each alternative routes are assume to be known *a priori* [16].

Summarizing the findings of the above discussion, no study has comprehensively dealt with remanufacturing job shop scheduling in subjecting to alternative recovery operations, random operation times and limited resource conflicts. In a practical remanufacturing system, the quality of returned products/cores varies, ranging from slightly used with minor blemishes to significantly damaged and requiring extensive repair. Such differences in the condition of products/cores strongly affect the set of recovery operations necessary to bring them up to a quality standard. In addition, the time that each recovery operation takes varies as well with respect to different damages of the cores. Those stochastic natures in remanufacturing triggers the potential conflicts of limited machine resources to a noticeable extend. Such characteristics have led a number of studies to investigate the operation management issues in remanufacturing environment by using different modeling methods, i.e., through analytic model [8,9,11,12,17], through Mixed Integer Programming model [10,13,15,16], through Lagrangian Relaxation [14] and through Petri nets model [18,19]. While our previous work [17] and [20] investigated the remanufacturing processes of worn cores with uncertain failure conditions and proposed an analytical model to study such dynamics in recovery process routings through probabilistic measures, how to incorporate those dynamic measures into remanufacturing scheduling has yet to be undertaken.

The colored timed Petri nets (CTPN) is an extension of ordinary Petri nets where the time and color are introduced to describe the logic structure and dynamic behavior of complex systems [21]. By taking advantage of both the well-formed formalism of CTPN and the robustness of Simulated Annealing (SA) for global optimization, the paper tackles this challenge and makes the following contributions. First, a colored timed Petri net is designed to explicitly model the dynamic of remanufacturing processes, such as various process routings, different uncertain operation times for cores, and real-time machine resource conflicts. With the color attributes in Petri nets, two types of decision points, recovery routing selection and resource dispatching, are introduced and embedded in places of CTPN model. With time attributes in Petri nets, the temporal aspect of recovery operations for cores as well as the evolution dynamics in cores' operational stages is mathematically analyzed. Second, a hybrid meta-heuristic algorithm embedded scheduling strategy over CTPN is proposed to search for the optimal recovery routings for worn cores and their recovery operation sequences on workstations, in minimizing the total production cost.

The rest of the paper is organized as follows. Section 2 presents the CTPN model. Section 3 describes the hybrid meta-heuristics based on SA and minimum slack time (MST) dispatching rule. A case study is conducted in Section 4, followed by the conclusion and future research in Section 5.

2. A CTPN based remanufacturing process model

2.1. Problem statement

This paper considers a typical remanufacturing job shop where its clients drop their used products with an expectation to get them fully recovered within a fixed time frame. The remanufacturing processes are traditionally complicated including disassembly that disassembles a worn product into components/cores, cores' recovery operations and assembly that assembles recovered cores back into the product. Figure 1 gives an organizational structure of the remanufacturing job shop.

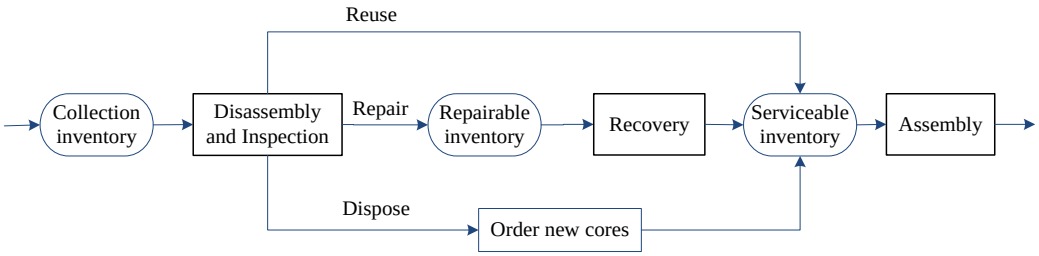

Figure 1. Organizational structure of the remanufacturing facility.

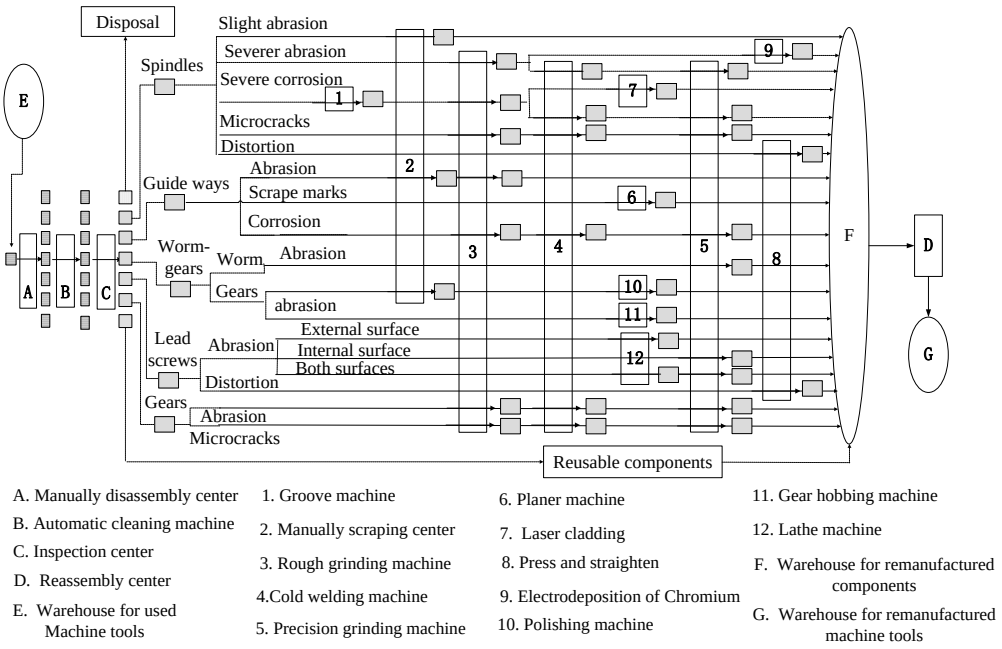

Figure 2. The remanufacturing process routings of machine tool components.

In this case, each used product (e.g., the l^{th} used product) arrives at the job shop with an arrival time $AT(l)$ and a due date $Due(l)$. The used product is first disassembled into what so called "cores" that are disassembled subassemblies or components. The cores are then classified into "reusable", "repairable", and "disposal" through the inspection process as depicted in Figure 1. This paper focuses on the recovery activities only. The challenge arises when dealing with the repairable cores with different levels of damage that need to go through a set of recovery operations to restore their size and property.

In a remanufacturing job shop, the quality of repairable cores commonly ranges from slightly used with minor blemishes to significantly damaged and requiring extensive recovery. Such quality variations strongly affect the set of recovery operations necessary to bring them up to a quality standard. As exemplified in Figure 2, there are several process routings for the remanufacturing of used machine tool cores of different types and various damage conditions. For instance, the lathe spindles can be inspected and classified into either "abrasion", "corrosion", or "micro-cracks", on which a set of specific recovery operations are designated for their remanufacturing.

With a wide variety of recovery technologies being applied into remanufacturing industry, some recovery process routings for cores with a certain type of damage can be alternative. For instance, if a spindle is detected with severe abrasion, two typical process routings would be "grinding → chromium electroplating" or "grinding → cold welding → fine grind" to regain the surface accuracy. Each alternative process routing consists of a set of predesigned recovery operations, with each operation being performed by a certain kind of machine equipment in a workstation, resulting in a specified recovery quality and unit processing cost [22,23]. As an upper level of process planning, different combinations of recovery process routings for cores would lead to much variation in remanufacturing cost and real-time machine workload in remanufacturing job shop.

With the recovery routings determined in the upper level, the core then travels to and is processed by a set of recovery workstations sequentially according to the designed process flow. However, the recovery operation time for a particular core in a given workstation is not fixed but rather dependent on the actual condition of the core. While different routings might demand operation times on the same workstation, various cores would compete for constrained machine resource to be recovered.

Our objective is to consider such variation for efficient resource utilization in fulfilling the demand with the minimum cost (i.e., operation cost and tardiness penalty). The remanufacturing job shop scheduling problem is strongly NP-hard due to: a) assignment decisions of reparable cores to recovery process routings and b) sequencing decisions of recovery operations of cores in each workstation. For that purpose, the paper makes the following assumptions:

(1) Each workstation has only one machine and each machine can process at most one task at a time.
(2) Each task is non-preemptive, requiring one and only one machine at a time.
(3) The transportation time between buffers is negligible.
(4) Penalty is charged if the system does not meet the due date of a product.

2.2. Definition of CTPN

The ordinary of Petri nets (PN) does not include time and color, which limits its capability to describe the logic structure and behavior of modeled systems, but not its evolution over time and

color [24,25]. In view of the specific characteristics of remanufacturing system to be modeled, this section introduces time and color attributes to a Petri net and proposes a CTPN. The model considers not only recovery process routings and stochastic operation times, but also integrates recovery routing assignments and efficient resource dispatching.

Definition 1: A colored timed Petri net (CTPN) is defined as 7-tuple: $CTPN = (P, T, I, O, M, U, C)$.

(1) $P = S \cup W$, where the set of places in S represents buffers, the set of places in W stands for workstations.

(2) $T = T^a \cup T^e$, where the finite set of transitions in T^a represent recovery operations, and the ones in T^e stand for transportation.

(3) I: $P \times T \rightarrow \{0,1\}$ is the input function that defines the set of ordered pairs (p_i, t_k), where $I(p_i, t_k)=1$, if p_i is an input place for t_k (i.e., $\forall p_i \in {}^{\bullet}t_k$), otherwise 0.

(4) O: $P \times T \rightarrow \{0,1\}$ is the output function that defines the set of ordered pairs (p_i, t_k), where $O(p_i, t_k)=1$, if p_i is an output place for t_k (i.e., $\forall p_i \in t_k^{\bullet}$), otherwise 0.

(5) M: $P \rightarrow \{0,1,2,\ldots\}$ is a marking vector, where $M(p_i)$ represents the number of tokens in p_i.

(6) U: $T^a \rightarrow \Theta$ is a non-zero time function that assigns to each recovery operation $t \in T^a$, while U: $T^b \rightarrow \Theta$ is a zero time function such that the transportation time between buffers is negligible.

(7) C: $P \rightarrow \Sigma$ is a color function that assigns to each place $p \in S \cup W$ with a color set $C(p)$.

Figure 3 gives an example of the proposed CTPN that corresponds to recovery process routings for used machine tools in Figure 2. The description for the places and transitions in CTPN are given Table 1. For the modeling and optimization purpose, some basic notations are introduced. Table 2 shows the index used in the CTPN model. Table 3 and Table 4 gives the model parameters and decision variables in the CTPN, respectively.

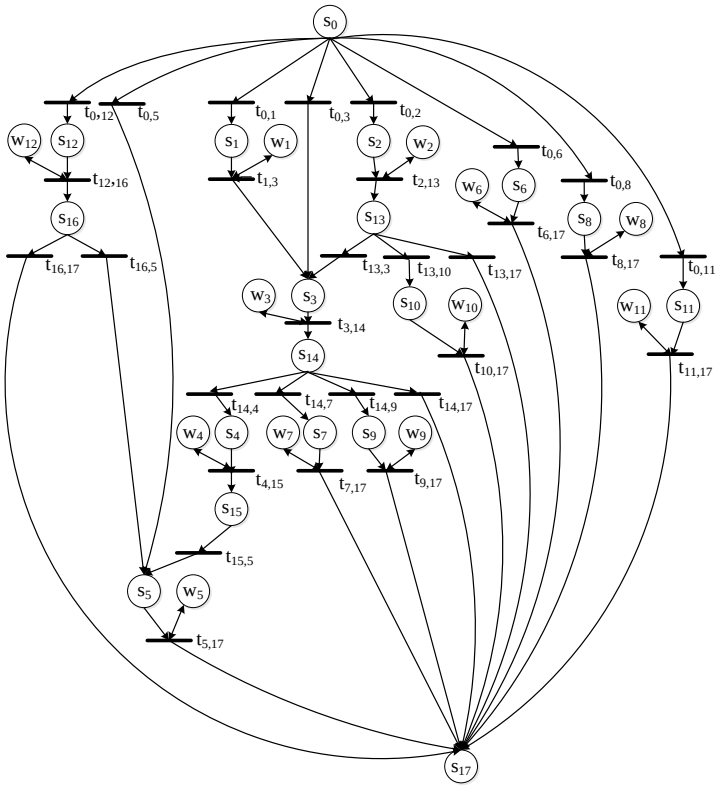

Figure 3. A CTPN for remanufacturing of machine tools in Figure 2.

Table 1. The description of places and transitions in CTPN.

Places and transitions in CTPN	Description
s_0	The output buffer of the inspection center
s_i (i=1,2,...,12)	The input buffer of each recovery workstation
s_z (z=13,14,15,16)	The output buffer of each recovery workstation
s_{17}	The input buffer of the reassembly center
w_k (k=1,2,...,12)	Workstation places
$t_{i,j}$ (i,j =1,2,...,12, $i\neq j$)	Recovery operations
$t_{0,i}$, $t_{i,z}$, $t_{i,17}$ (i=1,2,...,12, z=13,14,15,16)	Transportation transitions

Table 2. The index used in CTPN.

Index	Description
L= {1, 2, ..., L}	index set of used products
D= {1, 2, ..., D}	index set of component types
K= {1, 2, ..., K}	index set of workstations
H= {1, 2, ..., H}	index set of recovery process routings

Table 3. The parameters associated with the places and transitions in CTPN.

Parameters	Description
ϑ_{ld}	the d^{th} core disassembled from the l^{th} product
w_k	the k^{th} workstation that performs a specific recovery operation
R_h	the h^{th} recovery process routing designed for worn cores and is represented as a set of recovery operations
$pre(t_{ij}, R_h)$	the operational stages of recovery operation t_{ij} in R_h
Last(R_h)	the last recovery operation in R_h
a_{ldh}	a binary variable with a value 1 indicates the h^{th} routing for the core ϑ_{ld}
$u(t_{ij}, \vartheta_{ld})$	the time for a core ϑ_{ld} to be processed in transition t_{ij}
$AT(s_i, \vartheta_{ld})$	the arrival time of the core ϑ_{ld} at buffer place s_i
$ST(w_k, \vartheta_{ld})$	the start time of the core ϑ_{ld} being processed by w_k
$Com(w_k, \vartheta_{ld})$	the completion time of the core ϑ_{ld} processed by w_k
$\rho(\vartheta_{ld})$	the makespan of the component ϑ_{ld}
$\chi(\vartheta_{ld})$	the tardiness of the component ϑ_{ld}
$\zeta(l)$	the tardiness of the lth product
$Prob_{ld}$	the probability of a core ϑ_{ld} being selected for routing reassignment
$\Pi_{ld}^{h,x}$	the difference of the total average waiting time between the h^{th} routing and the x^{th} one for the core ϑ_{ld}
WT_k	the average waiting time per core consumed in the k^{th} workstation
Ψ_h	the total average waiting time per core consumed in the h^{th} routing
$V_{ld}^{h,x}$	the probability of the x^{th} routing being selected to replace the current h^{th} routing for the core ϑ_{ld}
$Slack_{xy}$	slack time of a core ϑ_{xy} in scheduling horizon

Table 4. The decision variables embedded in the places in CTPN

Decision variables	Description
r_{ldh}	a binary variable with a value 1 indicates that the hth process routing is assigned to a core ϑ_{ld}
$Pri(w_k, \vartheta_{ld})$	the priority for the d^{th} component of the l^{th} used product to be processed by the k^{th} workstation

2.3. Modeling remanufacturing processes with CTPN

(1) Modeling of recovery process routings

As stated earlier, the recovery process routing for a particular core is not deterministic but rather contingent on its damage level. When a core goes through inspection center and enter in the buffer place s_0, it will be assigned with a specified process routing (i.e., R_h), which is represented as a set of recovery operations that a core should go through sequentially for its recovery.

For the cores with alternative process routings to be recovered, decisions should be made on which "reasonable" process routings to be assigned to them. In the proposed CTPN model, the buffer place s_0 operates as a decision agent of choosing optimal recovery process routings for cores, by utilizing a hybrid scheduling algorithm elaborated in Section 3.

When the process routings for cores are determined, cores of different types and stochastic damage conditions require going through various recovery operations. In the CTPN model of Figure 3, different types of core tokens in s_0 might go to transitions among $t_{0,12}$, $t_{0,5}$, $t_{0,1}$, $t_{0,3}$, $t_{0,2}$, $t_{0,6}$, $t_{0,8}$, $t_{0,11}$; tokens in s_{13} may either go to $t_{13,3}$, $t_{13,10}$ or $t_{13,17}$; tokens in s_{16} might enters into either $t_{16,17}$ or $t_{16,5}$. In addition, cores of the same type and the same damage condition might go through different recovery processes. For example, when a spindle is detected with abrasion, its process routing can either be R_1 = { t_{03}, $t_{3,14}$, $t_{14,9}$, $t_{9,17}$} or R_2 ={ t_{03}, $t_{3,14}$, $t_{14,4}$, $t_{4,15}$, $t_{15,5}$, $t_{5,17}$}. Then in Figure 3, a token in s_{14} representing a spindle with abrasion might fire transition $t_{14,9}$ or $t_{14,4}$.

Thus, to make sure where the tokens in a buffer place should go, a unique color is introduced and associated with buffer places to distinguish the cores going through different recovery operations.

Definition 2: The color of a token ϑ_{ld} in a buffer place s_i is defined as $C(s_j, \vartheta_{ld}) = c_{ij}$ if the token goes to t_{ij} after being released from s_i.

Let $M(s_i)$ denote the number of tokens in place s_i regardless of token color, and $M(s_i, c_{ij})$ the number of tokens in place s_i at M that have the color c_{ij}. Therefore, $M(s_i) = \sum_j M(s_i, c_{ij})$.

Based on Definition 2, the color attached to a token in a buffer place changes through transition firings. Such color evolution represents the complex operational stages of a core according to its recovery process routing. For instance, if the recovery process routing of the core ϑ_{ld} is determined (i.e., r_{ldh} is known), its color evolution can then be derived as follows.

$$C(s_j, \vartheta_{ld}) = c_{jp}, \text{ if } \begin{cases} C(s_i, \vartheta_{ld}) = c_{ij} \\ pre(t_{jp}, R_h) = pre(t_{ij}, R_h) + 1, \ \forall r_{ldh} = 1 \\ s_i \in {}^{\bullet}t_{ij}, \ s_j \in {}^{\bullet}t_{jp} \end{cases} \tag{1}$$

(2) Modeling of shared resources

In a remanufacturing job shop, various cores would compete for constrained resource to be

recovered. An input buffer is then designated for each workstation in our CTPN model to accommodate all the cores that are waiting for processing. When multiple tokens occur in the input buffer place s_k of the kth workstation, decisions should be made on which core to be processed next as soon as the machine resource is ready. In the proposed CTPN model, each input buffer place s_i (i=1, 2, ...,12) is considered as a decision agent for resource dispatching and determines the operation sequences (priority) for the cores to be processed by the workstation.

Definition 3: The priority for the *core* ϑ_{ld} to be processed by the k^{th} workstation is defined as $Pri(w_k, \vartheta_{ld})$. The priority is a decision variable for scheduling optimization. The smaller the value is, the higher priority for the core to be processed by workstation w_k.

Definition 4: A transition t_{ij} in the CTPN developed above is enabled at a marking M to process the core ϑ_{ld} with a color c_{ij}, if and only if:

$$M(s_i, c_{ij}) \geq 1, \ \ s_i \in {}^{\bullet}t_{ij} \tag{2}$$

$$M(w_k) = 1, \ \ w_k \in {}^{\bullet}t_{ij} \cap t_{ij}^{\bullet} \tag{3}$$

$$Pri(w_k, \vartheta_{ld}) = \min(\ Pri(w_k, \vartheta_{xy})) \ \text{ and } \ C(s_i, \vartheta_{xy}) = c_{ij}, \forall x, \ y \tag{4}$$

Definition 5: An enabled transition t_{ij} can fire in a marking M with respect to a color c_{ij} yielding a new marking M' denoted as $M[(t_{ij}, c_{ij}) > M'$:

$$M'(s_i, c_{ij}) = M(s_i, c_{ij}) - 1, \ \ s_i \in {}^{\bullet}t_{ij} \tag{5}$$

$$M'(w_k) = M'(w_k), \ \ w_k \in {}^{\bullet}t_{ij} \cap t_{ij}^{\bullet} \tag{6}$$

Definition 6: When a transition t_{ij} is fired, the marking M' would transfer to M'' according to:

$$M''(s_i, c_{ij}) = M'(s_i, c_{ij}), \ \ s_i \in {}^{\bullet}t_{ij} \tag{7}$$

$$M''(w_k) = M'(w_k) + 1, \ \ w_k \in {}^{\bullet}t_{ij} \cap t_{ij}^{\bullet} \tag{8}$$

$$M''(s_j, c_{jp}) = M'(s_j, c_{jp}) + 1 \ \text{ and } \ C(s_j, \vartheta_{ld}) = c_{jp} \ \ \text{if} \begin{cases} s_j \in t_{ij}^{\bullet}, \ \ s_j \in {}^{\bullet}t_{jp} \\ pre(t_{jp}, R_h) = pre(t_{ij}, R_h) + 1 \end{cases} \tag{9}$$

According to the above transition enabling and firing rules, the system always dispatch an available workstation w_k to the core with the highest priority (i.e., the lowest $Pri(w_k, \vartheta_{ld})$ value) in the queue to maximize the system performance.

(3) Modeling of recovery operation times

When associating time with operation transitions, the CTPN model is able to describe the temporal aspect of recovery operations.

Definition 7: The time needed for a token ϑ_{ld} to be processed in transition t_{ij} is defined as $u(t_{ij}, \vartheta_{ld})$.

The processing times required to perform a necessary recovery operation is highly variable

since the quality and composition of returned products/parts varies. It is reported that the processing time data obtained from a remanufacturing facility follows an exponential distribution in order to simulate the large range of possible values [8]. The amount of processing time for a remanufacturing operation is widely modeled as an exponential function of the wear associated with a part [19]. In this section, we models the recovery operation time as the sum of average operation time and extended time, where the latter is a exponential function of the quality condition of the core being processed. To that end, quality testing of a core is conducted before recovery, on which an inspection score, $Score(\vartheta_{ld}) \in [0,1]$, is assigned to the core based. It is assumed that the better the quality, the higher the score and the shorter the operation time. So the relationship between the quality score and the processing time is modeled as follows.

$$u(t_{ij},\vartheta_{ld}) = \frac{-\ln(Score(\vartheta_{ld}))}{\beta(w_k)} + \lambda(w_k),\ t_{ij} \in w_k^{\bullet} \cap {}^{\bullet} w_k \tag{10}$$

where $\beta(w_k)$ is a control factor for workstation w_k whose value is determined based on statistics of historical data, and $\lambda(w_k)$ is the average operation time of cores in w_k. Whenever a component goes into a transition for processing, it must stay in transition t_j for more than $u(t_j, \vartheta_{ld})$ time units and then can be released into another buffer place.

(4) Temporal aspect of recovery operations

In the CTPN model, with the core tokens dynamically entering the buffer places, a set of transitions are then enabled and fired according to the enabling and firing rules. With the dynamic behavior of the CTPN, the remanufacturing operations are subject to the following constraints.

Operation start time requirements: Recovery operations cannot be started until the core has arrived at the buffer s_k of a corresponding workstation w_k. When the core arrived in the buffer s_k, it need waiting until all other cores with higher priorities are processed.

$$ST(w_k,\vartheta_{ld}) = \max\{AT(s_i,\vartheta_{ld}), Com(w_k,\vartheta_{ij})\},\ \forall Pri(w_k,\vartheta_{ld}) = Pri(w_k,\vartheta_{ij}) + 1, s_i \in {}^{\bullet} t_{ij}, t_{ij} \in {}^{\bullet} w_k \text{ I } w_k^{\bullet} \tag{11}$$

Processing completion time requirements: When a core goes into a workstation for processing, it must stay in a specific machine for more than $u(\vartheta_{ld}, w_k)$ time units and then can be released into the successor buffer.

$$Com(w_k,\vartheta_{ld}) = ST(w_k,\vartheta_{ld}) + u(t_{ij},\vartheta_{ld}),\ \forall t_{ij} \in w_k^{\bullet} \cap {}^{\bullet} w_k \tag{12}$$

Arrival time constraints: When a core is finished processing by a workstation, it can be transported immediately into another buffer of the successor workstation by ignoring the transportation time.

$$AT(s_i,\vartheta_{ld}) = Com(w_k,\vartheta_{ld}),\ \forall \begin{cases} r_{ldh} = 1 \\ \text{pre}(w_v,R_h) = \text{pre}(w_k,R_h) + 1 \\ s_i \in {}^{\bullet} t_{ij},\ t_{ij} \in {}^{\bullet} w_v \text{ I } w_v^{\bullet} \end{cases} \tag{13}$$

Using Eqs 11, 12 and 13 recursively, the makespan and tardiness of the component ϑ_{ld}, i.e., $\rho(\vartheta$ld) and $\chi(\vartheta_{ld})$, can be derived as shown in Eqs 14 and 15. The tardiness of the lth product is then calculated in Eq 16.

$$\rho(\vartheta_{ld}) = Com(w_k, \vartheta_{ld}) - AT(l),\ \forall Last(R_h) = t_{ij},\ t_{ij} \in w_k^{\bullet} \cap {}^{\bullet} w_k \tag{14}$$

$$\chi(\vartheta_{ld}) = \max\{0,\ \rho(\vartheta_{ld}) - Due(l)\} \tag{15}$$

$$\zeta(l) = \max_{d} \chi(\vartheta_{ld}) \tag{16}$$

3. CTPN-based hybrid meta-heuristics scheduler

Generally speaking, there are two ways to solve a scheduling problem by PN. One is to construct their reachability graph. However, it suffers from the state explosion problem [26]. Since we considered two types of decision variables in the proposed CTPN, using reachability graph for scheduling optimization might not be efficient. The other is to introduce scheduling algorithms into a PN model, which is adopted in this paper. The embedding style is optionally through places or transitions in a PN. This work uses the former mechanism and adopts two types of decision places in the proposed CTPN model. The decisions of choosing optimal recovery process routings for cores are embedded in buffer place s_0, while the input buffer place s_i (I = 1,2,...,12) makes decisions on dispatching machine resource in workstation to perform recovery operations for the cores that are waiting in queue. Considering the two types of scheduling decisions, a hybrid meta-heuristic algorithm using simulated annealing and minimum slack time rule is proposed and embedded in these decision places to search for global optimal process plans and schedules with a quick convergence speed.

The scheduling objective is to minimize both operation cost and tardiness penalty as shown in Eq 17. There are two types of decision variables: (1) recovery process routing for a given core (i.e., r_{ldh}), and (2) operation sequences for a given core in a workstation (i.e., $Pri(w_k, \vartheta_{ld})$).

Minimize:

$$TC = \sum_{l}\sum_{d}\sum u(t_{ij}, \vartheta_{ld}) \cdot \varphi(w_k) + \sum_{l} \zeta(l) \times \varpi,\quad \forall \begin{cases} r_{ldh} = 1 \\ t_{ij} \in R_h \text{ and } t_{ij} \in w_k^{\bullet} \end{cases} \tag{17}$$

where $\varphi(w_k)$ is unit operation cost of the k^{th} workstation and ϖ is unit penalty cost for product.

3.1. Simulated annealing for recovery process routing

SA is a stochastic gradient method for global optimization that has been applied to a wide variety of sophisticated combinatorial problems [27–29]. While it conducts local searches, it is capable of exploring the solution space stochastically to prevent from being trapped in a local optimum. Starting from an initial solution g_0, SA uses a certain mechanism to generate a neighborhood solution g' with an acceptance probability AP.

$$AP = \begin{cases} 1, & if\ \ \Delta < 0 \\ \exp(-\Delta / \Omega), & otherwise \end{cases} \tag{18}$$

where $\Delta = f(g') - f(g_0)$, Ω is called temperature, a global time-varying parameter, and $f(g)$ the optimization objective function. In this paper, SA is used to determine the optimal process routings for cores. At each iteration when a neighbor solution is generated, MST to be elaborated in section 3.2 is then applied to determine the operation sequences of the cores in each workstation. The parameter Ω is reduced by a factor η at each iteration and the chance of choosing an inferior solution decreases as well. The nested SA-MST search process continues until the stopping criterion is met.

In this section, the CTPN model is integrated and embedded with SA algorithm and MST dispatching rules to obtain the optimal remanufacturing routings and schedules. At each iteration of SA algorithm, the feasible recovery routing is first generated by the Initial/Neighborhood Solution Generation process. Then the CTPN model is established based on the arrival time of parts, the predetermined recovery routings and the processing time of each operations. Whenever multiple parts arrived in a same buffer place in the CTPN, MST rules is used for resource dispatching. The CTPN model evolves to a final version until the resource conflicts in all the buffer places are tackled by the SMT. Finally, the operation cost and tardiness penalty of a SA solution at each iteration can be calculated according to the time aspects marked in the CTPN, and the next iteration of SA algorithm goes on until the stopping criterion is met. The flowchart of the CTPN-based hybrid meta-heuristics scheduler is shown in Figure 4.

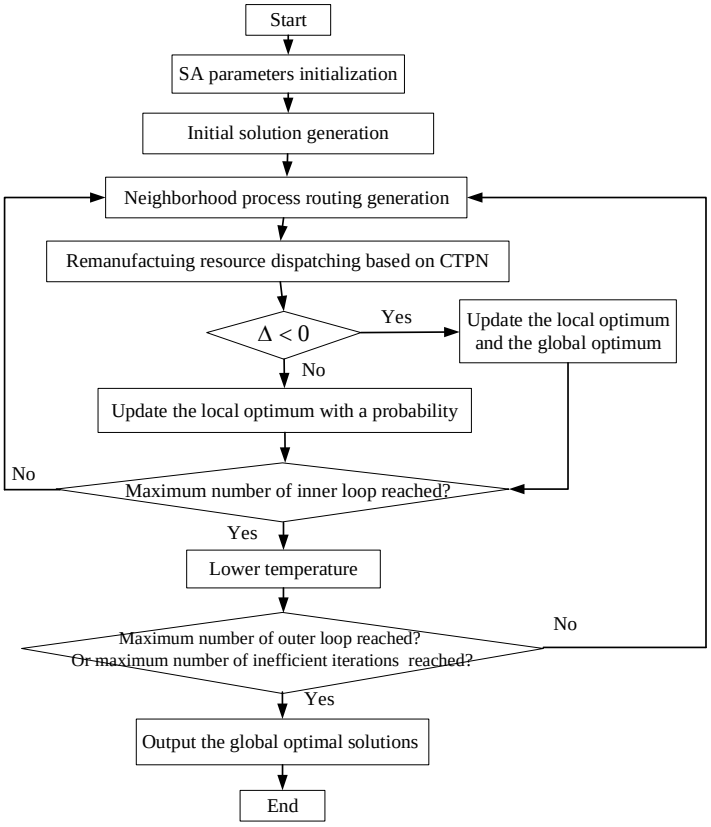

Figure 4. The flowchart of CTPN-based hybrid meta-heuristics scheduler.

(1) SA solution representation

In this paper, a 3-dimension matrix $R = [r_{ldh}]_{L\times D\times H}$ is proposed as representation of SA solutions. Each element r_{ldh} is a binary variable, one indicating the h^{th} recovery process routing is assigned to

the core ϑ_{ld} and zero otherwise. Given that not all routings are feasible to a specific core, the viability of a randomly generated R is controlled via a logic AND operation in Eq 19:

$$R = R \wedge \mathrm{A} = [r_{ldh} \wedge a_{ldh}] \tag{19}$$

where $A = [a_{ldh}]$ is a L-by-D-by-H mask whose element $a_{ldh} = 1$ indicates the feasibility of the h^{th} routing for the core ϑ_{ld}.

(2) Initial solution generation

The choice of an initial solution is vital to the performance of the algorithm [30]. To that end, a cost function of a routing (i.e., the h^{th} routing) for a core ϑ_{ld} is defined in Eq 20. The smaller the value of q_{ldh}, the greater chance the h^{th} routing is selected for the core, i.e., $r_{ldh} = 1$.

$$q_{ldh} = \frac{\sum_{j} u(t_{ij}, \vartheta_{ld}) \times \varphi(w_k)}{\sum_{h}\sum_{j} u(t_{ij}, \vartheta_{ld}) \times \varphi(w_k)}, t_{ij} \in R_h, r_{ldh} = 1, t_{ij} \in^{\bullet} w_k \ \mathrm{I} \ w_k^{\bullet} \tag{20}$$

(3) Neighborhood solution generation

Traditional SA randomly generates a neighborhood solution g' from the current solution g by using a predefined move mechanism [31,32]. In this section, a multiple moves mechanism is used for neighborhood solution generation [33,34]. That is, for each current solution, randomly selects an element in the matrix R and uses one type of moves (i.e., ς-opt, $\varsigma \in \{1, 2, \ldots, H\text{-}h\}$) to exchange r_{ldh} with $r_{ld[h+\varsigma]}$ for a new neighborhood solution.

To improve the convergence of SA, this work opts out the random selection process. Instead, at each annealing step, the proposed SA identifies cores with the potential to improve their routing assignment and to minimize their tardiness. To that end, $\chi(\vartheta_{ld})$ is used to evaluate the urgency of routing reassignment for cores. The probability of a core being selected for routing reassignment is defined as follows.

$$prob_{ld} = \chi(\vartheta_{ld}) \Big/ \sum_{l}\sum_{d} \chi(\vartheta_{ld}) \tag{21}$$

For any identified core, the selection of an alternative yet better routing is determined based on an efficiency index, $\Pi_{ld}^{h,x}$, which calculates the difference of the total average waiting time between the current process routing (i.e., the h^{th} routing) and the alternative one (i.e., x^{th} routing):

$$\Pi_{ld}^{h,x} = \Psi_h - \Psi_x, \forall \begin{cases} r_{ldh} = 1 \\ x \in \{1,2,\Lambda\ ,H\} - \{h\} \\ r_{ldx} = 0,\ \ a_{ldx} = 1 \end{cases} \tag{22}$$

$$\Psi_h = \sum_{t_{ij} \in R_h, w_k \in^{\bullet} t_{ij}} WT_k \qquad \Psi_x = \sum_{t_{ij} \in R_x, w_k \in^{\bullet} t_{ij}} WT_k \tag{23}$$

$$WT_k = \frac{\sum_{l}\sum_{d} (ST(w_k, \vartheta_{ld}) - AT(s_i, \vartheta_{ld}))}{N_k} \tag{24}$$

where WT_k is the average waiting time per core consumed in the k^{th} workstation, Ψ_h and Ψ_x are the total average waiting time per core consumed in the h^{th} and x^{th} process routing, respectively, and N_k is the total number of cores being processed by workstation w_k. When the core ϑ_{ld} is switched from the current routing to the other one with a larger $\Pi_{ld}^{h,x}$, it is more likely to relieve resource conflict and accomplish the task with a greatly reduced tardiness. Therefore, the probability of the x^{th} routing being selected to replace the current h^{th} routing for the core is defined as

$$v_{ld}^{h,x} = \Pi_{ld}^{h,x} \Big/ \sum_{\substack{x\in\{1,2,\Lambda,H\}-\{h\} \\ a_{ldx}=1}} \Pi_{ld}^{h,x} \tag{25}$$

(4) Stopping criterion

In this proposed algorithm, two thresholds are pre-defined: the maximum number of inefficient iterations where the global optimum is not updated I_{max}, and the maximum number of outer loop iterations ol_{max}. Our SA will stop whenever either of the thresholds is reached.

3.2. *Remanufacturing resource dispatching based on CTPN*

With a SA solution, each core follows the derived process routing to be recovered step by step. When multiple cores enter a workstation and compete for the same machine resource, a secondary optimization is conducted to determine the operation sequences of cores to be processed by the workstation, aiming to minimize the total tardiness penalty. The dispatching rules have been widely used for operation sequence optimization to address job shop scheduling problems [35]. Minimum Slack Time (MST), as one of typical heuristic sequencing rules, is widely used for the scheduling problems. MST measures the "urgency" of a job by its slack time, which is defined as the difference between its due date and production time. That is, the shorter the slack time, the higher priority to be processed. In the case of MST dispatching rules, the jobs might be processed as soon as possible in order to be finished right before its due date. As a result, the tardiness of all the jobs can be minimized. Recently, MST rule have been widely used for operation sequence optimization and are embedded with some meta-heuristic approaches to address various scheduling problems [36,37]. In this section, MST is chosen for resource dispatching and is embedded with SA algorithm to search for the optimal schedules in minimizing both the operation cost and the penalty cost with related to tardiness.

For a specific core, the slack time of its recovery operation can be calculated as the temporal difference between the due date, the current time, and the remaining operation times, as formulated in Eq 26. MST sequences jobs in the descending order of slack time. Whenever a workstation w_k releases a component ϑ_{uv} ($\forall Pri(w_k, \vartheta_{uv}) = q > 0$) and turns back into idle state, it will pick up a core ϑ_{ld} with the least slack time ($\exists\ slack_{ld} = \min\{slack_{xy}\}$) from the waiting queue ($\forall \vartheta_{xy}$, $AT(s_i, \vartheta_{xy})$-$Com(w_k, \vartheta_{ld}) \le 0$), and then immediately process it in the $(q + 1)^{\text{th}}$ sequence, as shown in Eq 27.

$$slack_{xy} = Due(l) - Com(w_k, \vartheta_{uv}) - \sum_{pre(t_{cd},R_h)\ge pre(t_{ij},R_h)} u(t_{cd}, \vartheta_{xy}),\ \forall r_{xyh} = 1,\ t_{ij} \in R_h,\ t_{cd} \in R_h,\ t_{ij} \in^{\bullet} w_k \text{ I } w_k^{\bullet},\ Pri(w_k, \vartheta_{uv}) = q \tag{26}$$

$$Pri(w_k, \vartheta_{ld}) = q+1,\ if\ \begin{cases} AT(s_i, \vartheta_{xy}) - Com(w_k, \vartheta_{uv}) \le 0,\ \forall r_{uvh} = 1, t_{ij} \in R_h, s_i \in^{\bullet} t_{ij},\ t_{ij} \in^{\bullet} w_k \text{ I } w_k^{\bullet},\ Pri(w_k, \vartheta_{uv}) = q \\ slack_{ld} = \min\left\{slack_{xy}\right\}, \forall r_{xyh} = 1, t_{ij} \in R_h \end{cases} \tag{27}$$

The solution obtained by MST is a 3-dimension matrix $G = [g_{ldk}]_{L\times D\times K}$, where g_{ldk}, a non-negative integer, represents the operation sequence of the core ϑ_{ld} to be performed by workstation w_k. $g_{ldk} = 0$ means that the core ϑ_{ld} is not processed by w_k. Once the matrix G is derived, $Pri(w_k, \vartheta_{ld})$ in CTPN are determined and thereby the total cost can be calculated.

With the recovery process routing R and operation sequence matrix G being derived through the proposed hybrid meta-heuristics, a set of transitions are then be enabled and fired as core tokens dynamically enter or release from the buffer places.

4. Case study

To fully understand the above concepts and algorithms, the remanufacturing of a batch of obsolete machine tools in an example system is used to demonstrate the remanufacturing scheduling. The workflows of cores in this remanufacturing shop and its corresponding CTPN are shown in Figs. 2 and 3. In order to verify the efficiency of the proposed method, three cases are considered for simulation. In the baseline case, cores are recovered through a set of fixed process routings. An available workstation takes MST rule for resource dispatching. In the second case, the traditional SA uses a standard *neighborhood generation method* and is embodied with MST rule for routing assignment and resource dispatching. In the third case, the recovery routing for a core and the resource dispatching for a workstation are guided through the proposed SA/MST by using the proposed *neighborhood solution generation method*.

All the algorithms are implemented in Matlab 2009. The simulation running duration is set to be 30 days. Everyday's work hours are 24 hours. Data of the first 3 days are thrown off as warm-up consideration. For the simulation, the input data is populated as follows.

(1) Since that the machine resources in the floor shop are fixed and limited, the machine workloads and the intension of resource conflicts increases with the rise of arrival rate, and *vice versa*. In order to simulated a set of resource conflict scenarios, the random arrival of used machine tools follows a Poisson distribution with an arrival rate per hour $\pi \in \{5, 6, 7, 8, 9, 10, 11\}$. The designed arrival rate can limit the machine workloads (i.e., the average length of waiting queue on the machines) in the floor shop within a satisfactory range [0.75, 3.6].

(2) The due date (in days) for each used machine tool is randomly generated by a uniform distribution, $Due(l) \sim U[5, 10]$.

(3) Each core disassembled from used machine tools is inspected before its recovery and is assigned a quality score $Score(\vartheta_{ld})$ through an Exponential distribution, $Score(\vartheta_{ld}) \sim \Gamma(1, \tau)$ where $\tau \in [0.08, 0.10]$.

(4) The control factor and average operation time for each workstation is set to $\beta(w_k) = 0.2$ and $\lambda(w_k) = 15$.

(5) The operation cost per hour in workstations 7 and 9 (i.e., chromium electroplating and laser cladding) is \$100, while that in other workstations is \$50.

(6) The penalty cost per day per machine tool is set to be $\varpi \in \{20, 60, 100\}$ in dollars.

(7) The parameters in SA are set to $ol_{\max} = 5000$, $il_{\max} = 100$, $I_{\max} = 30$, $\Omega = 169340$, $\eta = 0.998$.

(8) Considering that the total number (L) of machine tools generated in each run of the simulation varies, our comparison focuses on normalized objective values. In particular, the comparison evaluates the total cost per product (*tc*), operation cost per product (*pc*), tardiness penalty per product (*dc*), and the average waiting time per core consumed in each workstation (*wt*).

Our first set of experiments evaluates how the three methods respond to system demand and various penalty cost rates. In each trial, we run the three methods independently. It should be noted that the baseline case only employs MST rule for resource scheduling, while the proposed SA/MST and the standard one run continuously until they meet the stopping criteria. As the arrival rate of used machine tools changes from 5 to 11, the average total cost *tc* is obtained and compared among the three methods. This process is repeated with different penalty cost rates (i.e., ϖ = 20, 60, and 100) as shown in Tables 5, 6, and 7, respectively.

Table 5. Comparison of the algorithms performance under different arrival rate (ϖ = 20).

Arrival rate	The average total cost per machine tool			Improvement of the standard method over the baseline one	Improvement of the proposed method over the standard one
	Baseline case tc_1	Standard SA/MST tc_2	Proposed SA/MST tc_3	tc_1–tc_2	tc_2–tc_3
π=5	186.66	117.02	117.02	69.64	0
π=6	228.37	133.48	118.21	94.89	15.27
π=7	293.05	172.45	146.89	120.6	25.56
π=8	335.87	208.06	168.69	127.81	39.37
π=9	447.52	266.47	216.25	181.05	50.22
π=10	581.63	352.28	270.91	229.35	81.37
π=11	714.29	461.86	335.77	252.43	126.09
Sample mean	398.20	244.52	196.25	153.68	48.27
Sample standard deviation	193.44	125.46	82.67	68.93	43.15

Table 6. Comparison of the algorithms performance under different arrival rate (ϖ = 60).

Arrival rate	The average total cost per machine tool			Improvement of the standard method over the baseline one	Improvement of the proposed method over the standard one
	Baseline case tc_1	Standard SA/MST tc_2	Proposed SA/MST tc_3	tc_1–tc_2	tc_2–tc_3
π=5	215.76	117.83	117.83	97.93	0
π=6	327.61	160.31	140.12	167.3	20.19
π=7	455.9	226.47	185.02	229.43	41.45
π=8	658.09	342.57	273.22	315.52	69.35
π=9	891.23	515.96	406.38	375.27	109.58
π=10	1137.96	738.41	559.67	399.55	178.74
π=11	1530.54	986.65	775.72	543.89	210.93
Sample mean	745.30	441.17	351.14	304.13	90.03
Sample standard deviation	472.28	324.23	244.76	151.99	80.25

Table 7. Comparison of the algorithms performance under different arrival rate (ϖ = 100).

Arrival rate	The average total cost per machine tool			Improvement of the standard method over the baseline one	Improvement of the proposed method over the standard one
	Baseline case tc_1	Standard SA/MST tc_2	Proposed SA/MST tc_3	tc_1–tc_2	tc_2–tc_3
π=5	270.84	117.32	117.32	153.52	0
π=6	463.08	224.8	186.58	238.28	38.22
π=7	687.42	416.46	275.13	270.96	141.33
π=8	1063.8	655.68	453.47	408.12	202.21
π=9	1495.67	957.36	706.49	538.31	250.87
π=10	1917.52	1231.56	918.76	685.96	312.8
π=11	2403.39	1483.89	1120.77	919.5	363.12
Sample mean	1185.96	726.72	539.79	459.24	186.94
Sample standard deviation	789.17	517.36	385.07	273.81	135.61

The performance of the standard SA/MST and the baseline one is first evaluated and compared by calculating the cost difference under different arrival rates, as shown in Figure 5. It is clear that the proposed SA/MST outperforms the baseline method in terms of generating less cost, particularly in reacting to the larger arrival rates. This is reasonable since that when a substantial amount of cores are fighting for limited resources for to be recovered, the standard SA/MST operates on routings reassignment of cores to reduce the ever intensified resource conflicts and thus minimize a relatively large amount of tardiness penalty to lower the total cost, while the baseline method is not capable of addressing so by fixing process routings of cores. The results in Figure 5 can be also interpreted as evidence that the hybrid SA/MST is promising for remanufacturing scheduling optimization.

The differences of total cost per product obtained by the two hybrid methods are also calculated and plotted in Figure 6. The following remarks can be made regarding the performance of the proposed SA/MST over the standard one.

(1) When the arrival rate is low (e.g., π = 5), the system has sufficient enough resources to handle remanufacturing demand. Thus the benefit of our proposed SA/MST in comparison to the standard one is minor. As the system demand increases with the rising arrival rate, more and more used machine tools are fighting for the limited resource to be remanufactured. The proposed method then outperforms the standard method through a more targeted annealing process to properly reassign recovery routings of cores and resolve ever intensified resource conflict. Therefore, the advantage of the proposed method over the standard one becomes significant.

(2) In a similar fashion, by comparing the corresponding row in Tables 5, 6 and 7, the relative improvement of the proposed method over the standard one becomes more and more significant with the increasing penalty cost. Using π = 8 as an example, the improvement increases from 11% when ϖ = 20, to 17% when ϖ = 60, and to 30% when ϖ = 100.

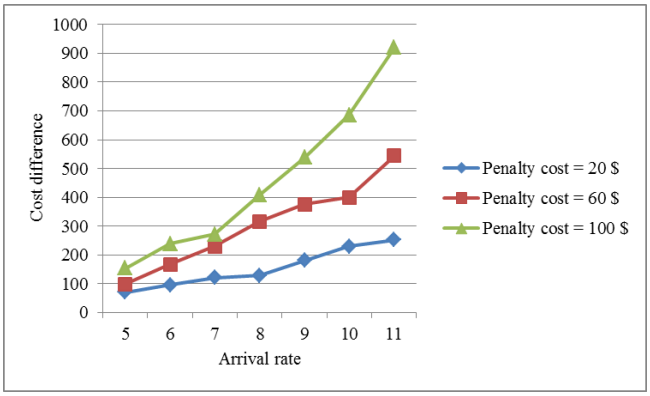

Figure 5. The improvement of the standard sa/mst over the baseline case.

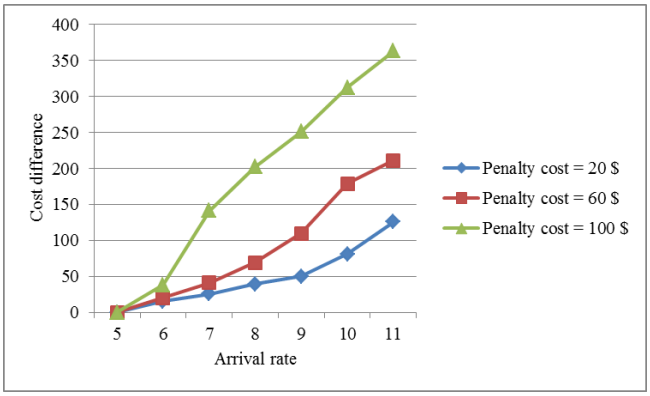

Figure 6. The improvement of the proposed SA/MST over the standard SA/MST.

To further show the significant difference between the performances of the two hybrid algorithms, a sample mean, a sample standard deviation and confident interval estimation are introduced. A relative value $\sigma = -(tc_3 - tc_2)/tc_2$ is defined to characterize the improvement of the best solution obtained by the proposed SA/MST over the one obtained by the standard one. A 95% confidence interval of the improvement rate σ is conducted using the data in Tables 5–7. The approximate $100(1-\alpha)\%$ confidence interval for σ is defined as

$$\begin{aligned} &\bar{\sigma} \pm t_{1-\alpha/2}(n-1)\frac{SD}{\sqrt{n}} \\ or\ \ &\bar{\sigma} - t_{1-\alpha/2}(n-1)\frac{SD}{\sqrt{n}} \le \sigma \le \bar{\sigma} + t_{1-\alpha/2}(n-1)\frac{SD}{\sqrt{n}} \end{aligned} \tag{28}$$

where $\bar{\sigma}$ is the sample mean of σ based on a sample of size n; SD is the sample standard deviation; $t_{1-\alpha/2}(n-1)$ is the $100(1-\alpha)\%$ percentage point of a t-distributed with $n-1$ degree of freedom. The 95% confidence intervals for the improvement σ under three scenarios (i.e., ϖ = 20, 60 and 100) are given in Eqs 29, 30 and 31, respectively. It is obvious that the 95% confidence interval for σ lies completely above zero (i.e., $\bar{\sigma} > 0$), which provides strong evidence that the proposed SA/MST is better than the standard one in terms of generating less total cost.

$$0.0814 \le \hat{\sigma}_1 \le 0.2455 \tag{29}$$

$$0.1042 \le \hat{\sigma}_2 \le 0.3118 \tag{30}$$

$$0.1414 \le \hat{\sigma}_3 \le 0.3911 \tag{31}$$

The results in Tables 5–7 represents significant enhancement of the proposed methodology in terms of overall total cost per product, especially when the system is over utilized and its responsiveness to fluctuating market demand is important.

Table 8. Comparison of the algorithm performance over time (π = 8 and ϖ = 60).

Computation time breaks	Standard SA/MST				Proposed SA/MST			
	tc_1	pc_1	dc_1	wt_1	tc_2	pc_2	dc_2	wt_2
After 0sec	399.25	112.16	287.09	76.80	399.25	112.16	287.09	76.80
After 30sec	380.38	116.78	263.60	72.37	371.14	114.62	256.51	55.21
After 60sec	393.70	113.71	279.99	61.76	341.42	118.56	222.86	48.34
After 90sec	393.70	113.71	279.99	58.90	311.93	120.51	191.43	46.49
After 120sec	386.11	115.62	270.49	58.75	292.79	121.75	171.04	44.54
After 150sec	385.56	119.96	265.60	58.42	280.06	123.28	156.78	41.52
After 180sec	384.17	121.18	262.99	57.90	274.14	124.64	149.51	41.47
After 210sec	383.35	122.86	260.49	56.51	273.33	126.31	147.03	38.40
After 240sec	378.22	124.65	252.57	54.73	273.22	132.79	140.42	35.67
No time limits*	342.57	129.41	213.16	48.07	273.22	132.79	140.42	35.67

* "No time limit" means the simulation keeps the algorithm running until it meets the stopping criteria elaborated above.

In the second set of experiments, the performance of the two hybrid algorithms is further evaluated by calculating the normalized objective values achieved after pre-specified time breaks (Table 8), where the arrival rate and the penalty cost are set as π = 8 and ϖ = 60. The following remarks can be made regarding algorithms' performance.

(1) As the computation time increases, both algorithms strive to search for a better scheduling solution that maintains a good trade-off between operation cost and tardiness penalty. It should be noted that both algorithms start at the same initial solution (i.e., zero time break). Since the initial solution is generated with the goal of minimizing operation cost only, it presents a relatively larger penalty cost. The annealing step in both methods then begins to rearrange routings for cores that ultimately balances the overall workload of workstations and thus reduces waiting time to lower the tardiness penalty. As shown in Table 8, at the 210-second time break, the optimal solutions generated by both methods have a bit larger *pc* and a quite smaller *dc*, resulting in a much lower total cost per product.

(2) The proposed SA/MST method converges much rapidly than the standard one, providing a better solution within a given short time frame. This is intuitively understandable since the proposed SA/MST uses resource conflict and tardiness penalty to guide neighborhood solution generation while the random mechanism in the standard SA fails to address so, resulting in a relatively larger tardiness penalty.

All simulation output analysis verifies the effectiveness of the proposed hybrid SA/MST algorithm for the remanufacturing scheduling.

5. Conclusion

Remanufacturing systems are faced with a greater degree of uncertainty and complexity than traditional manufacturing systems, leading to the need for planning and control systems designed to deal with the added uncertainty and complexity. Although many new methodologies have been led up to deal with various operation management issues in remanufacturing environments, no study has comprehensively dealt with remanufacturing job shop scheduling in subjecting to alternative recovery operations, random operation times and limited resource conflicts. A CTPN is introduced to model the dynamics of remanufacturing process, such as various process routings, uncertain operations times and resource conflicts. With time and color attributes in PN, two types of decision variables are linked with places in CTPN and the evolution of system dynamics in recovery operations are mathematically analyzed. With the support of the computation model via CTPN, a hybrid meta-heuristic using SA and MST rule is proposed and embedded with CTPN to sample large search space efficiently and search for a good scheduling solution in minimizing total production cost. The performance of the proposed SA/MST algorithm is compared against another two cases: baseline case with fixed recovery process routings and case 2 using standard SA/MST. The comparison results provide strong evidence that the proposed scheduling method is of significant importance in achieving minimum production cost especially when the system is overloaded and its responsiveness to shared resource conflicts is important.

Our research can be also extended in several directions. For instance, the integration of alternative recovery routing selection and resource dispatching is a NP-hard problem. Therefore, it is worth of investigation to take the structure advantage of CTPN for a more efficient optimization heuristics. Another challenge is to take into consideration of unexpected system disruptions (i.e., random machine breakdown) in remanufacturing scheduling. The future work also includes further validation of our methodology using more factory data.

Acknowledgement

This work was supported in part by the Fundamental Research Funds for the Central Universities (SWU22300503), the National Natural Science Foundation of China (51875480) and the National Natural Science Foundation of China (51475059).

Conflict of interest

All authors declare that there is no conflict of interests in this paper.

References

1. G. D. Li, M. Reimann and W. H. Zhang, When remanufacturing meets product quality improvement: The impact of production cost, *Eur. J. Oper. Res.*, **271** (2018), 913–925.

2. P. V. Loon and L. N. V. Wassenhove, Assessing the economic and environmental impact of remanufacturing: a decision support tool for OEM suppliers, *Int. J. Prod. Res.,* **56** (2017), 1662–1674.
3. M. Matsumoto, K. Chinen and H. Endo, Remanufactured auto parts market in Japan: Historical review and factors affecting green purchasing behavior, *J. Clean Prod.,* **172** (2018), 4494–4505.
4. B. M. Liu, D. J. Chen and W. J. Zhou, The effect of remanufacturing and direct reuse on resource productivity of China's automotive production, *J. Clean Prod.,* **194** (2018), 309–317.
5. Y. F. Zhang, S. C. Liu and Y. Liu, The 'Internet of Things' enabled real-time scheduling for remanufacturing of automobile engines, *J. Clean Prod.,* **185** (2018), 562–575.
6. J. Zhou and Q. W. Deng, An environmental benefits and costs assessment model for remanufacturing process under quality uncertainty, *J. Clean Prod.,* **186** (2018), 180–190.
7. R. Kumar and P. Ramachandran, Revenue management in remanufacturing: perspectives, review of current literature and research directions, *Int. J. Prod. Res.,* **54** (2016), 2185–2201.
8. V. D. R. Guide, R. Srivastava and M. E. Kraus, Priority scheduling policies for repair shops, *Int. J. Prod. Res.,* **38** (2000), 929–950.
9. V. D. R. Guide, G. C. Souza and E. V. D. Lann, Performance of static priority rules for shared facilities in a remanufacturing shop with disassembly and reassembly, *Eur. J. Oper. Res.,* **164** (2005), 341–353.
10. R. H. Teunter, K. Kaparis and O. Tang, Multi-product economic lot scheduling problem with separate production lines for manufacturing and remanufacturing, *Eur. J. Oper. Res.,* **191** (2008), 1241–1253.
11. S. Zanoni, A. Segerstedt, O. Tang, et al., Multi-product economic lot scheduling problem with manufacturing and remanufacturing using a basic period policy, *Comput. Ind. Eng.,* **62** (2012), 1025–1033.
12. H. Sun, W. D. Chen, B. Y. Liu, et al., Economic lot scheduling problem in a remanufacturing system with returns at different quality grades, *J. Clean Prod.,* **170** (2018), 559–569.
13. M. G. Kim, J. M. Yu and D. H. Lee, Solution algorithms for scheduling flow-shop-type remanufacturing systems. The 14th Asia Pacific Industrial Engineering and Management System Conference Vietnam, December 8–11, (2013).
14. P. B. Luh, D. Q. Yu, S. Soorapanth, et al., Relaxation based approach to schedule asset overhaul and repair services, *IEEE T. Autom. Sci. Eng.,* **2** (2005), 145–157.
15. H. J. Wen, S. W. Hou, Z. H. Liu, et al., Integrated lot sizing and energy-efficient job shop scheduling problem in manufacturing/remanufacturing systems, *Chaos Solitons Fractals,* **105** (2017), 69–76.
16. R. Zhang, S. K. Ong and A. Y. C. Nee, A simulation-based genetic algorithm approach for remanufacturing process planning and scheduling, *Appl. Soft. Comput.,* **37** (2016), 521–532.
17. C. B. Li, Y. Tang, C. C. Li, et al., A modeling approach to analyze variability of remanufacturing process routing, *IEEE T. Autom. Sci. Eng.,* **10** (2013), 86–89.
18. S. E. Zhao, Y. L. Li and R. Fu, Fuzzy reasoning Petri nets and its application to disassembly sequence decision-making for the end-of-life product recycling and remanufacturing, *Int. J. Comput. Integr. Manuf.,* **27** (2014), 415–421.
19. Y. Tang, M. C. Zhou and M. M Gao, Fuzzy-Petri-net-based disassembly planning considering human factors, *IEEE T. Syst. Man Cybern. Syst.,* **36** (2006), 718–726.

20. L. L. Li, C. B. Li and Y. Tang, An integrated approach of reverse engineering aided remanufacturing process for worn components, *Robot. Comput. Integr. Manuf.,* **48** (2017), 39–50.
21. D. H. Wu and W. Zheng, Formal model-based quantitative safety analysis using timed Coloured Petri Nets, *Reliab. Eng. Syst. Saf.,* **176** (2018), 62–79.
22. H. L. Liao, Q. W. Deng and Y. R. Wang, An environmental benefits and costs assessment model for remanufacturing process under quality uncertainty, *J. Clean Prod.,* **178** (2018), 45–58.
23. G. D. Li, M. Reimann and W. H. Zhang, When remanufacturing meets product quality improvement: The impact of production cost, *Eur. J. Oper. Res.,* **271** (2018), 913–925.
24. J. Y. Sheng and D. Prescott, A hierarchical coloured Petri net model of fleet maintenance with cannibalisation, *Reliab. Eng. Syst. Saf.,* **168** (2017), 290–305.
25. S. A. Hussain, N. A. Khan and A. Sadiq, Simulation, modeling and analysis of master node election algorithm based on signal strength for VANETs through Colored Petri nets, *Neural Comput. Appl.*, **29** (2018), 1243–1259.
26. Y. W. Si, V. I. Chan, M. Dumas, et al., A Petri nets based generic genetic algorithm framework for resource optimization in business processes, *Simul. Model. Pract. Theory,* **86** (2018), 72–101.
27. A. Assad and K. Deep, A Hybrid Harmony search and Simulated Annealing algorithm for continuous optimization, *Inf. Sci.*, **450** (2018), 246–266.
28. Z. Y. Liu, Z. S. Liu, Z. P. Zhu, et al., Simulated annealing for a multi-level nurse rostering problem in hemodialysis service, *Appl. Soft. Comput.,* **64** (2017), 148–160.
29. F. Erchiqui, Application of genetic and simulated annealing algorithms for optimization of infrared heating stage in thermoforming process, *Appl. Therm. Eng.*, **128** (2018), 1273–1272.
30. X. Y. Li, C. Lu, L. Gao, et al., An effective multi-objective algorithm for energy efficient scheduling in a real-life welding shop, *IEEE T. Ind. Inform.,* **14** (2018), 5400–5409.
31. S. Hore, A. Chatterjee and A. Dewanji, Improving variable neighborhood search to solve the traveling salesman problem, *Appl. Soft. Comput.,* **68** (2018), 83–91.
32. X. Y. Li, L. Gao, Q. K. Pan, et al., An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop, *IEEE T. Syst. Man. Cybern. Syst.,* (2018).
33. Y. Z. Zhou, W. C. Yi, L. Gao, et al., Adaptive differential evolution with sorting crossover rate for continuous optimization problems, *IEEE T. Cybern.,* **47** (2017), 2742–2753.
34. X. Y. Li and L. Gao, An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *Int. J. Prod. Econ.*, **174** (2016), 93–110.
35. G. R. Amin and A. El-Bouri, A minimax linear programming model for dispatching rule selection, *Comput. Ind. Eng.*, 121 (2018), 27–35.
36. P. Neammanee and M. Reodecha, A memetic algorithm-based heuristic for a scheduling problem in printed circuit board assembly, *Comput. Ind. Eng.*, 56 (2009), 294–305.

37. B. N. Silva, M. Khan and K. Han, Load balancing integrated least slack time-based appliance scheduling for smart home energy management, *Sensors*, 18 (2018).

AIMS Press